Network Covert Channels on the Android Platform

[Extended Abstract]

Wade C. Gasior
University of Tennessee at Chattanooga
615 McCallie Avenue
Chattanooga, TN 37403
wade-gasior@utc.edu

Li Yang
University of Tennessee at Chattanooga
615 McCallie Avenue
Chattanooga, TN 37403
li-yang@utc.edu

ABSTRACT

Network covert channels are used to exfiltrate information from a secured environment in such a way that an observer cannot detect that communication is taking place at all. These secret channels have been identified as an important security threat to governments and the private sector, and several research efforts have focused on the design, detection, and prevention of such channels in enterprise-type environments. Mobile devices such as smartphones and tablets have become an ubiquitous computing platform, and are storing or have access to an increasingly large amount of sensitive information. As such, these devices have become prime targets of attackers with malicious intents. This paper discusses the implementation of network covert channels on the Android mobile platform, and shows that data can be leaked from these devices in a manner undetectable by the user, the phone's security features, or network security between the mobile device and the outside network. Understanding the threat of covert channels to mobile devices will allow the development of proactive protection mechanisms.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; H.1.1 [**Models and Principles**]: Systems and Information Theory—*Information Theory*; D.4.6 [**Operating Systems**]: Security and Protection—*Information flow controls*

General Terms

Security

Keywords

Security, Mobile, Android, Covert Channels

1. INTRODUCTION

A network covert channel is the use of a shared resource, namely a network communications channel, to transfer information in a way which it was not initially designed for. Network covert channels are used to exfiltrate data from a secure location to a non-secure location without being detected or prevented by traditional information security protection techniques such as firewalls, encryption, or intrusion detection systems.

This paper discusses the implementation of network covert channels on the Google Android mobile platform. Mobile devices such as smart phones and tablets have become an ubiquitous computing platform, and an increasing amount of sensitive data is being stored on these portable devices. The Android platform accounts for just under 50% of the worldwide smart phone market [9], and combined with its open application market policy, is a prime target for malicious applications that steal user's data. This paper will show that covert communication channels can be implemented on the Android platform that allow data to be leaked from the device to a remote server in a manner where it appears that no subversive communication is taking place.

Mobile platforms currently have only limited implementations of firewalls, intrusion detection systems, and other network security features, but this is likely to change as the information value that these devices hold increases.

Gaining a better understanding and developing improved methods of network covert channel prevention and detection are vital to the information security efforts in both private and government sectors, and have been the focus of much research in the past years. Little, if any, of this research has explored covert channels on mobile platforms. It is important that research begin in this area in order to develop proactive protection and prevention mechanisms.

2. BACKGROUND AND RELATED WORK

2.1 Problem Description

Covert channels can be described using the analogy of two prisoners attempting to escape. Simmons proposed the 'prisoner problem' in 1983, which is the standard model used when describing covert channel communication [13]. The model describes two people, Alice and Bob, who are imprisoned and intend to escape. In order to agree upon an escape plan, they must communicate, but a third party (Wendy the Warden) monitors their communications. If Wendy detects anything suspicious, she throws the two into solitary confinement, cutting off all communication and making their escape impossible. In order to hatch an escape plan, Alice and Bob must exchange messages that appear innocent, but contain hidden information that Wendy will not notice.

This scenario is translated to a network environment [8] where Alice and Bob use two networked computers for communication (and Alice and Bob may be the same person). Wendy represents the network management that exists between Alice and Bob (firewalls, intrusion detection systems, etc.) that are able to monitor the traffic between Alice and Bob and alter, disrupt, or eliminate a detected subversive communication channel.

2.2 Purpose

The primary goal of a covert channel is to hide the fact that communication is taking place at all. Compare this to cryptography, where the primary goal is to transfer data that is only readable by the receiver [2].

Covert channels are desirable to exfiltrate sensitive information for several reasons. One, the use of a normal communications channel (such as an FTP or HTTP connection) is easily detected by wardens looking for malicious traffic. This type of traffic can be captured in log files and traffic dumps, and then analyzed and prevented. Making the communication channel more obscure, by methods such as using nonstandard port numbers, is also easily detectable and would trigger mechanisms such as packet anomaly detection systems [2].

Our goal is to show that communication channels between an Android device and a remote server can be implemented in ways that are undetectable by network wardens.

2.3 Design of Covert Channels

Network covert channels have been implemented in a number of ways, all designed with goals of resistance to detection and increased capacity. In a broad sense, network covert channels can be classified as either storage-based or timing-based, but the distinction between the two is quite blurred.

Storage-based network covert channels operate by altering the content of some resource that can be observed by a receiver [5]. Cabuk describes the implementation of a simple binary storage-based covert channel in [5]. This channel operates using a timeline divided into intervals of size t, known by both the sender and receiver. The sender transmits a bit value of 0 by maintaining silence throughout a given interval, and transmits a bit value of 1 by sending a packet or becoming active during a given interval.

Timing-based network covert channels operate by altering the delays between network events, such as the sending of a packet. Berk implemented one of the earlier examples of a network covert timing channel [2]. This channel operates using a set of time intervals $t_1, t_2, ..., t_n$. Each time interval is associated with a symbol from the input alphabet, and the delay between consecutive packets (inter-packet delay) is altered to transmit a given symbol. More recent work has focused on creating more robust network covert channels that are difficult to detect.

Cabuk developed the idea of time-replay covert channels [3]. This implementation uses a sample of inter-packet delays from legitimate traffic S_{in} partitioned into equal sized bins $S_0, S_1, ..., S_n$. To transmit a symbol I_n, a random inter-packet delay from S_n is transmitted.

Gianvecchio developed model-based covert timing channels [7]. This approach effectively evades detection by modeling and mimicking the statistical properties of legitimate traffic. First, a filter monitors legitimate traffic of the type to be mimicked (e.g. ftp), collecting the inter-packet delays. Second, an analyser fits the inter-packet delays in sets to legitimate traffic using max likelihood estimation (MLE). Finally, an encoder generates random inter-packet delays to be used for the covert channel based on the best-fit model. This type of covert channel is very difficult to detect if the normal traffic being modelled is independent identically distributed. A similar model-based approach was developed by Sellke et al. and is discussed in [11].

2.4 Detection of Covert Channels

Efforts towards the detection of covert channels have primarily focused on timing-based channels. The goal of detection, given a chain of consecutive delays between packets ΔT_i, is to determine if it is possible to affirm with a certain probability that there has been malicious intent from within the network [2]. In other words, are the observed delay times specific or discernible enough to most likely **not** have been generated by chance?

There are two general types of detection techniques: regularity-based [4, 12] and shape-based [2, 1]. The regularity of traffic is described by second or higher-order statistics, such as correlations in the data. Regularity tests include variance testing and other statistical methods. The shape of traffic is described by first-order statistics including mean, variance, and distributions. Shape tests include the Kolmogorov-Smirnov Test and entropy-based detection.

Cabuk discusses early efforts towards developing regularity-based tests in [4] by examining variance patterns in a set of observed inter-packet delays. This method is implemented by separating inter-packet delays from observed traffic into "windows" of size w. Then, for each window i, the standard deviation σ_i is computed. Next, the difference between each pair of windows (i, j) is computed. Finally, the standard deviation of all differences is computed to obtain a summary statistic. This can be expressed in the formula:

$$regularity = STDEV(\frac{|\sigma_i - \sigma_j|}{\sigma_i}, i < j, \forall i, j) \qquad (1)$$

Covert channels will have a smaller regularity value. This test is effective at detecting very simple covert channels, but fails with more advanced implementations such as model-based channels [7].

Berk developed an early detection technique based on the determining the Shannon capacity of a channel (how many bits per symbol yields the highest transmission rate). The technique assumes that an attacker would want to transmit information in the most efficient means possible. Observed traffic is analyzed to see if it matches a channel that would be implemented using the encoding scheme producing the highest channel capacity. This detection technique was effective only at detecting channels that prioritized speed over stealth.

Gianvecchio developed a detection technique using the Kolmogorov-Smirnov test, which determines whether or not two samples (or a sample and a distribution) differ. The test measures the maximum distance between two emperical distribution functions:

$$KSTEST = \max |S_1(x) - S_2(x)| \tag{2}$$

where S_1 and S_2 are empirical distribution function of the two samples. Gianvecchio employed this technique with some success to measure the difference between known legitimate traffic and observed traffic to determine if a covert channel exists [6].

Gianvecchio also developed an entropy-based detection technique [6]. Entropy is a measure of the amount of information missing if the value of a variable is unknown. Entropy will be low if the value of the variable can be predicted with a greater certainty (e.g. a loaded coin flip), and higher if the variable cannot be as easily predicted (e.g. a fair coin flip). Covert traffic is more regular, and thus has a lower entropy than legitimate traffic. This technique is highly effective at detecting covert channels.

3. COVERT CHANNELS ON ANDROID

3.1 Overview

To evaluate network covert channels on the Android platform, we designed two implementations that stealthily transmit data from the phone to a remote server: a timing-based channel and a storage-based channel. Our timing channel is encoded using delays between network events, while our storage channel is encoded based on the ordering of network events.

3.2 Challenges

During our implementations of network covert channels on the Android platform, we were met by three primary challenges.

One challenge we faced in our research was that of accessing the targeted sensitive data on the Android device. The Android operating system requires applications to request fine-grained permissions from the user (e.g. permission to access the network, and permission to access the contacts list). Applications that have both network access and access to sensitive data such as the contacts list raise suspicions from the user who must approve these permissions, and from security software that identifies over-privileged applications [10]. Exasperating this problem is the fact that applications on the Android platform are executed in isolated running environments. The Android operating system is a Linux-based OS where each application is run as a distinct user and group, which creates a sandboxed environment that separates each application from one another and from the underlying system. This prevents dividing the application into multiple applications with separate security privileges in an attempt to avoid raising suspicion.

A second challenge of our research involved the specific implementation of the network covert channels. The Android platform does not allow an application to have access to raw packets, so techniques used in traditional computing environments that alter the flow of packets in order to implement covert channels [2, 14, 6, 3] are not possible. The lowest level of network access available on Android is the use of raw sockets (exposed through Java libraries).

A third challenge faced was the difficulty of implementing timing-based covert channels over cellular networks. Mobile phone networks introduce a large amount of jitter, and are very inconsistent which causes synchronization and reliability issues in our covert channels. We also discovered that very few UDP packets complete the trip from source to destination without being dropped.

To overcome the challenge of accessing sensitive data on the Android platform, we intend to use the method of on-device covert storage channels discussed by Schlegel et al [10]. Schlegel showed that the protection mechanisms on the Android platform that prevent unauthorized application-to-application communication can be subverted. The implementation puts in place malware in the form of two cooperating applications: one with permission to access sensitive data, and the other with network access. On-device covert channels were used to communicate data between the two applications. These channels relied primarily on changing a global setting that both applications had access to, such as the device's volume. Currently, due to development time constraints, we assume that our covert channel applications have access to the desired sensitive data on the target device while also having full network access, and thus do not actually implement Schlegel's technique, but rather assume that a technique such as this would be employed by a real world application.

To overcome the challenge of the lack of low level packet access, we implemented custom TCP and UDP protocols to carry our legitimate application traffic. We encode a covert timing channel within the legitimate channel by altering the delays between protocol messages.

The traffic quality challenges presented by the cellular network have forced us to use the TCP protocol only, since UDP traffic often does not make it through the network. We are also forced to use multiple delays to represent a single input symbol due to the differences in transmitted and observed delays. This results in a low bandwidth covert channel. We are currently working on improved synchronization mechanisms and the implementation of error correcting algorithms in our timing-based channel to attempt to improve the bandwidth of the covert channel.

3.3 Experimental Setup

For our research, we used a Motorola Droid X running Android 2.3 on the Verizon 3G network in Chattanooga, TN. The Droid X has a 1 GHz Texas Instruments IMAP3630 processor and 512 MB RAM. As a remote destination, we used a server on the University of Tennessee at Chattanooga campus, approximately 25 network hops away from the device. Our applications are implemented using the Java language and the Android development kit.

We designed two innocuous appearing applications that transmit the contacts list from the device to our remote server by way of a covert channel.

For the timing channel implementation, we required an application that would generate a large amount of legitimate traffic at a steady pace to make the embedding of a covert timing channel possible. To accomplish this, we developed an application that transmits live video from the camera on the Android device to the remote server. The application allows the phone to operate as an IP camera which generates a large amount of legitimate traffic. Two simple protocols were implemented to transmit the video from device to server: one utilizing UDP and one utilizing TCP. As mentioned above, many UDP packets do not make it through the Verizon network and to the remote destination, and thus TCP was the only successful implementation. The TCP implementation works by transmitting video frame by frame. The covert channel is implemented by altering the delays between video frames sent to the server. Currently, we use two delays (one long and one short). We encode the contacts list into binary form, and then transmit a zero by transmitted two short delays, and a one by transmitting two long delays. We signify the beginning and end of a covert channel transmission by transmitting a short-long-short-long delay sequence. In future work, we plan to implement more advanced forms of covert channels such as model-based channels.

The server side of this application displays video streamed from the device while capturing and recording the delays between received frames. The application reconstructs the transmitted contacts list from the observed delays and displays the data as it is received in plain text. The server side application was written in Java and runs on a standard Linux installation.

To implement a storage-based covert channel, we designed an application that displays a small advertisement banner at the bottom of an arbitrary application. The ads are fetched from our remote server, of which there are n choices of advertisements.

The application leaks information by requesting a specific advertisement to represent a specific encoded input token (binary values of the contacts list). If 2^n advertisements are available, then each request can represent n bits of the data to be transmitted. The application fetches ads using http requests with a POST parameter representing the specific ad to be fetched. Ads of certain values represent the beginning or end of a covert transmission.

The server side of this application responds to http requests with the appropriate ad, and records the sequence in which ads are requested during a covert transmission. The application displays covert data in plain text as it is received.

4. CONCLUSION

Our work is an early example demonstrating that it is possible to design applications that exfiltrate sensitive data from the Android platform in a way that is extremely difficult to detect. The currently proposed covert channel detection and prevention mechanisms are not suitable for mobile platforms, and would most likely not be reasonable to implement on a network-wide basis. Therefore, new mechanisms of detection and prevention applicable to the types of security risks discussed in this paper must be developed.

5. REFERENCES

[1] V. Berk, A. Giani, and G. Cybenko. Covert channel detection using process query systems. 2005.

[2] V. Berk, A. Giani, and G. Cybenko. Detection of Covert Channel Encoding in Network Packet Delays. Technical Report TR2005-536, Dartmouth College, Computer Science, Hanover, NH, August 2005.

[3] S. Cabuk. *Network Covert Channels: Design, Analysis, Detection, and Elimination*. PhD thesis, Purdue University, 12 2006.

[4] S. Cabuk, C. E. Brodley, and C. Shields. Ip covert timing channels: design and detection. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 178–187, New York, NY, USA, 2004. ACM.

[5] S. Cabuk, C. E. Brodley, and C. Shields. Ip covert channel detection. *ACM Trans. Inf. Syst. Secur.*, 12:22:1–22:29, April 2009.

[6] S. Gianvecchio and H. Wang. An entropy-based approach to detecting covert timing channels. *IEEE Transactions on Dependable and Secure Computing*, 99(PrePrints), 2010.

[7] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia. Model-based covert timing channels: Automated modeling and evasion. In *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, RAID '08, pages 211–230, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] T. G. Handel and M. T. Sandford, II. Hiding data in the osi network model. In *Proceedings of the First International Workshop on Information Hiding*, pages 23–38, London, UK, 1996. Springer-Verlag.

[9] B. Reed. Android market share nears 50% worldwide. `http://www.networkworld.com/news/2011/080111-canalys.html`, 2011.

[10] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, pages 17–33, Feb. 2011.

[11] S. H. Sellke, C.-C. Wang, S. Bagchi, and N. B. Shroff. Tcp/ip timing channels: Theory to implementation. In *INFOCOM*, pages 2204–2212. IEEE, 2009.

[12] G. Shah, A. Molina, and M. Blaze. Keyboards and covert channels. In *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, Berkeley, CA, USA, 2006. USENIX Association.

[13] G. J. Simmons. The prisoners' problem and the subliminal channel. In *CRYPTO*, pages 51–67, 1983.

[14] S. Zander, G. Armitage, and P. Branch. A survey of covert channels and countermeasures in computer network protocols. *Communications Surveys Tutorials, IEEE*, 9(3):44 –57, quarter 2007.